ETL Process Documentation

This document details the Extract, Transform, Load (ETL) process for the GBC_Superstore database. It includes step-by-step instructions with example SQL scripts and Python code.

---

1. Extraction

1. **Connect to the Source Database and/or Files**
    - Use SQLAlchemy in Python to connect to the MySQL database (GBC_Superstore).
    - Extract data from the normalized tables (orders, order_details, shipping, customers, customer_address, postal_codes, regions, products, sub_categories, and categories).

Example SQL Query for Data Extraction:

```
SELECT
    o.order_id,
    o.order_date,
    o.order_return_status,
    od.sales,
    od.quantity,
    od.discount,
    od.profit,
    r.region_name,
    pc.city,
    cat.category_name,
    sc.sub_category_name,
    p.product_id,
    p.product_name,
    o.ship_date
FROM orders o
JOIN order_details od ON o.order_id = od.order_id
JOIN shipping s ON o.ship_mode_id = s.ship_mode_id
JOIN customers c ON o.customer_id = c.customer_id
JOIN customer_address ca ON c.customer_id = ca.customer_id
JOIN postal_codes pc ON ca.postal_code = pc.postal_code
JOIN regions r ON pc.region_id = r.region_id
JOIN products p ON od.product_id = p.product_id
JOIN sub_categories sc ON p.sub_category_id = sc.sub_category_id
JOIN categories cat ON sc.category_id = cat.category_id;
```

2. **Python Code for Extraction:**

```
import pandas as pd

from sqlalchemy import create_engine

engine = create_engine('mysql+pymysql://root:@127.0.0.1:3306/GBC_Superstore', echo=False)
sql_query = """<above SQL query>"""
df = pd.read_sql(sql_query, con=engine)
df['order_date'] = pd.to_datetime(df['order_date'])
df['year_month'] = df['order_date'].dt.to_period('M')
```

---

2. Transformation

1. **Data Cleaning and Type Conversion**
   - Convert date fields to datetime format.
   - Clean column names and fill any missing values as required.
2. **Derive Additional Fields**
   - Create new columns such as order_date_day, quarter, and year for grouping.
   - Calculate Key Performance Indicators (KPIs):
     - **Profit Margin (%)**: (Total Profit / Total Sales) * 100
     - **Sales Growth (%)**: Compute day-over-day or month-over-month changes.
     - **Year-over-Year Change (%)** for the executive report.
3. **Aggregation Examples Using Python (Pandas):**

Operational Report (Daily):

```
df['order_date_day'] = df['order_date'].dt.date
group_cols_op = ['order_date_day', 'region_name', 'city', 'category_name',
'sub_category_name']
df_op = df.groupby(group_cols_op).agg(
    total_sales=('sales', 'sum'),
    quantity_sold=('quantity', 'sum'),
    discounts_applied=('discount', 'sum'),
    total_profit=('profit', 'sum')
).reset_index()
df_op['profit_margin(%)'] = (df_op['total_profit'] / df_op['total_sales']) * 100
```

Executive Report (Monthly/Yearly):

```
group_cols_ex = ['region_name', 'year_month']

df_ex = df.groupby(group_cols_ex).agg(
```

```
    total_sales=('sales', 'sum'),
    total_profit=('profit', 'sum'),
    total_discount=('discount', 'sum'),
    orders_count=('order_id', 'nunique')
).reset_index()
df_ex['profit_margin(%)'] = (df_ex['total_profit'] / df_ex['total_sales']) * 100
# Compute sales growth per region
df_ex = df_ex.sort_values(['region_name','year_month'])
df_ex['prev_sales'] = df_ex.groupby('region_name')['total_sales'].shift(1)
df_ex['sales_growth(%)'] = ((df_ex['total_sales'] - df_ex['prev_sales']) / df_ex['prev_sales'] *
100).fillna(0)
```

---

3. Loading

1. **Load Transformed Data into Target Database Tables**
    - Create temporary tables to store report data if needed.
    - Use SQL scripts to create the table, and Python code to insert the transformed data.

    Example SQL Script to Create the Executive Report Table:

```
DROP TABLE IF EXISTS executive_report;

CREATE TABLE executive_report (
    region_name VARCHAR(100),
    year_month VARCHAR(20),
    total_sales DECIMAL(10,2),
    total_profit DECIMAL(10,2),
    profit_margin DECIMAL(10,2),
    sales_growth DECIMAL(10,2),
    top_performing_product VARCHAR(255),
    discount_impact DECIMAL(10,2),
    return_rate DECIMAL(10,2),
    average_order_value DECIMAL(10,2)
);
```

2. **Python Code to Insert Data into the Executive Report Table:**

```
from sqlalchemy import text


# Assume df_executive_report is the final transformed DataFrame
df_executive_report = df_ex[['region_name', 'year_month', 'total_sales', 'total_profit',
```

```
                    'profit_margin(%)', 'sales_growth(%)', 'discount_impact',
'orders_count']]

# Rename columns to match table schema
df_executive_report = df_executive_report.rename(columns={
    'profit_margin(%)': 'profit_margin',
    'sales_growth(%)': 'sales_growth'
})

with engine.begin() as conn:
    conn.execute(text("DROP TABLE IF EXISTS executive_report"))
    conn.execute(text("""
        CREATE TABLE executive_report (
            region_name VARCHAR(100),
            year_month VARCHAR(20),
            total_sales DECIMAL(10,2),
            total_profit DECIMAL(10,2),
            profit_margin DECIMAL(10,2),
            sales_growth DECIMAL(10,2),
            discount_impact DECIMAL(10,2),
            orders_count INT
        )
    """))
    records = df_executive_report.to_dict(orient='records')
    for rec in records:
        conn.execute(text("""
            INSERT INTO executive_report (
                region_name, year_month, total_sales, total_profit, profit_margin,
                sales_growth, discount_impact, orders_count
            ) VALUES (
                :region_name, :year_month, :total_sales, :total_profit, :profit_margin,
                :sales_growth, :discount_impact, :orders_count
            )
        """), **rec)
```

3. **Export Data to CSV**
    - Once data is loaded into the table, query the table and export its content to a
      ```
      df_final = pd.read_sql("SELECT * FROM executive_report", engine)

      df_final.to_csv("executive_report_table.csv", index=False)
      ```

---

4. Automation and Scheduling

1. **Automate the ETL Process**
    - Use cron jobs (Linux) or Task Scheduler (Windows) to run the ETL script at scheduled intervals (daily, weekly, or monthly).
2. **Logging and Monitoring**
    - Implement logging within the Python script to capture success or failure messages.
    - Monitor the output files and database tables for data quality and consistency.

Conclusion

This ETL process extracts data from the GBC_Superstore source, transforms it through cleaning, aggregation, and KPI calculation, and then loads it into target database tables. The process also generates final reports (operational and executive) that include subtotals, totals, titles, units, and trend comparisons.